**UNIVERSITY OF MICHIGAN SCHOOL OF INFORMATION**
**SI 539: Design of Complex Web Sites**

**Assignment 8 – Don't Repeat Yourself**

---

**Due Date:** Sunday March 30, 2008 at 11:55PM

In this assignment you will use many of the Rails features to reduce the amount of repetition in your code. You will start to see some of the benefits of the Rails framework. Also you will add some view functionality and delete functionality and explore some Rails features and functionality.

This assignment description gives very direct instructions - explanation of the new features that your application is using will be in the corresponding lecture for this material.

Before starting this - Make a copy of your Assignment 7 and do the work described in Assignment 7A and test it thoroughly.

For those of you who are doing your own applications

**Refactor the Markup Using Partial Rendering and an Application Template**

Our first task is to remove the repetition of the navigation in each of the .rhtml files in your application.

Create a file called:

**app\views\one_nav.html**

Put this in it:

```
<div id="navigation">
  <ul>
    <li><a href="<%= url_for :action => "index" %>">About</a></li>
    <li><a href="<%= url_for :action => "contact" %>">Contact</a></li>
    <li><a href="<%= url_for :action => "pictures" %>">Pictures</a></li>
    <li><a href="<%= url_for :action => "members" %>">Membership</a></li>
    <li><a href="<%= url_for :action => "join" %>">Application</a></li>
  </ul>
</div>
```

This is a partial render file that is a fragment of HTML that can be reused throughout your application using partial rendering.

Next we will create a application-wide template - this template will be used to "wrap" the rest of your rhtml views. The "yield" line below indicated where in the template the other view files will be included.

Create a file called:

**app\views\layouts\application.rhtml**

And place this code in it:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
   <title>Assignment 8 - SI539</title>
   <meta http-equiv="Content-Type"
        content="text/html; charset=utf-8" />
   <%= stylesheet_link_tag "style.css" %>
</head>
<body>
   <div id="header">
     <h1>Welcome to SI539</h1>
   </div>
   <%= render :partial => 'nav' %>
   <div id="bodycontent">
      <%= yield %>
   </div>
</body>
</html>
```

Then go through all of your view files and remove all the material that has been moved into the application-wide template. For example the file **contact.rhtml** looks as follows:

```
<h2>Welcome to the SI539 Sample Site</h2>
<p>For more information, send mail to
<a href="mailto:info@si539.com">info@si539.com</a>.</p>
```

Effectively it only contains the content between the bodycontent divs - I moved the bodycontent divs into the **application.rhtml** file.

Once this is complete test your application for functionality - make sure to view the source of each page to make sure that you only have one DOCTYPE and <head>, etc. Make sure that the including is working of both the application-wide template and the navigation code.

Nicely, things like stylesheets and titles are exactly one place so making changes can all be done one place.

Note that the class="selected" is gone - so you won't know what page you are on - we will fix this next.

**Building your Rails Helpers for your Templates**

We need to solve the problem of getting class="selected" onto the right element of the list. Depending on which action you are in, a different list member is selected.

We are going to make our own helper to generate the <li> entry. Edit the file

**app\helpers\one_helper.rb**

Make is look as follows (there is a lot of Ruby in this file):

```
module OneHelper
  # Note that the action we are in is stored in params[:action]

  def do_nav_entry(textparam, actionparam)
    logger.info "In do_select text=#{textparam} action=#{actionparam}"

    seltext = ''
    if actionparam == params[:action]
      logger.info "We found the matching action!"
      seltext = ' class="selected"'
    end

    # get the URL for our action
    urltext = url_for(:action => actionparam)
    logger.info "urltext=#{urltext}"
    return '<li><a href="' + urltext + '"'+ seltext+'>'+textparam+"</a></li>"
  end

end
```

Theno modify this file

**app\views\one_nav.rhtml**

Replace each of the <li> lines. Old Line:

**<li><a href="<%= url_for :action => "index" %>">About</a></li>**

New Line:

**<%= do_nav_entry("About","index") %>**

Do this for all of the navigation lines in the _nav.rhtml file.

Test your application - now the current page should appear properly selected as you move from page to page. If you have any problems - view the source of the page and see what kind of bad things are happening in your HTML.

At this point you have removed a lot of repetition in your views.

### Viewing and Deleting Members

The next step is to create a series of screens that allow you to view and/or delete members.

Edit the members.rhtml and add another column to the table with a heading titled "Action" and put this code into that column in each of the member rows:

```
<td>
<%= link_to 'View', { :action => 'view', :id => valencia },
      :method => :post %> /
<%= link_to 'Delete', { :action => 'delete', :id => valencia} ,
     :confirm => 'Are you sure?', :method => :post %></td>
```

Add the following methods to your controller:

```
def delete
  memb = Member.find(params[:id])
  memb.destroy()
  flash[:notice] = "User Deleted"
  redirect_to :action => 'members'
end

def view
  @memb = Member.find(params[:id])
end
```

Edit the **members.rhtml** file and add the following:

```
<p>These are the members of SI539 who are in good standing and
  have paid all their dues.</p>
<% if flash[:notice] %><p class="notice"><%= flash[:notice] %></p><% end %>
<table>
```

Create a new view file

**app\views\one\view.rhtml**

And add the following code to the file:

```
<h2>Member Detail</h2>
<p>
<strong>Name:</strong>
<%= @memb.name %>
</p>
<p>
<strong>Email:</strong>
<%= @memb.email %>
</p>
```

Note that there is no head, body, or even div id="bodycontent" - this all comes from the application wide layout.

Then edit the **join.rhtml** file adding the code in bold below.

```
<h2>Joining SI539 Alumni Group</h2>
<% if flash[:notice] %><p class="notice"><%= flash[:notice] %></p><% end %>
<form method="post" action="<%= url_for :action => "thanks" %>">
```

Then edit the controller file and modify the method as indicated by the bold information below:

```
def thanks
    if params[:yourname] == nil or params[:yourmail] == nil or
     params[:yourname] == "" or params[:yourmail] == ""
```

```
        flash[:notice] = "Please specify both name and E-Mail"
        redirect_to :action => 'join'
        return
      end
      memb = Member.create()
      memb.name = params[:yourname]
      memb.email = params[:yourmail]
      memb.save
      @barcelona = memb.id
  end
```

This insures that both fields in your form muse be filled in and you will get an error if the data is not entered into the fields.

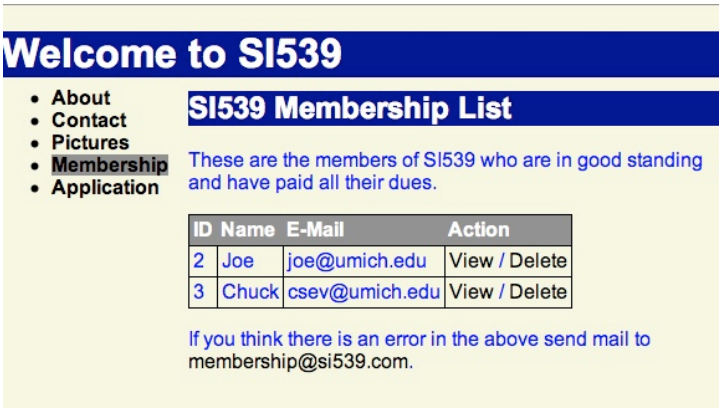
Add an element in your style sheet that makes the notice class be shown in a red font.

**Test Plan**

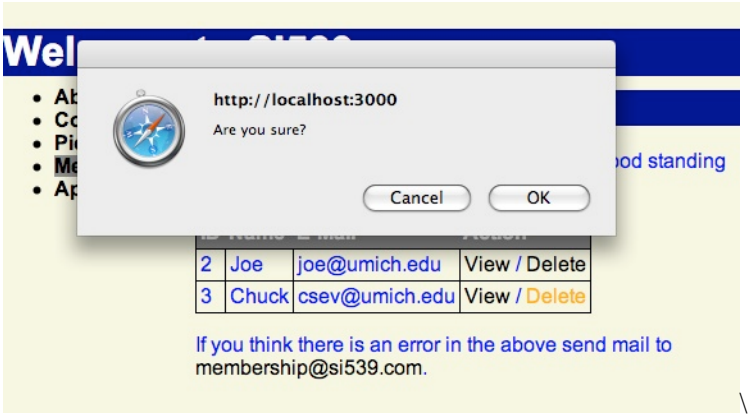
You must pass the following tests to after you have completed the assignment.

All pages must render properly with the current page properly highlighted just like the previous assignment.

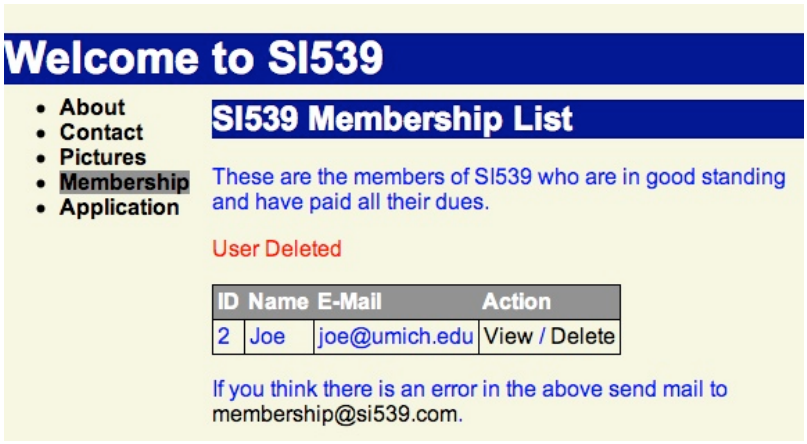
The membership list should have the new View/Delete option as follows:



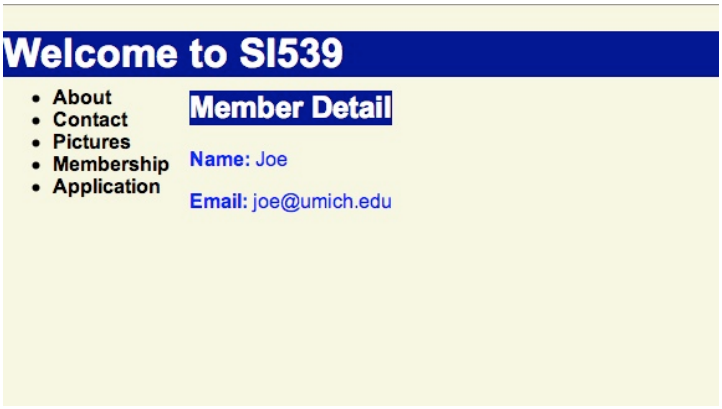

When you press Delete - a confirmation should pop up - it should work for OK and Cancel.



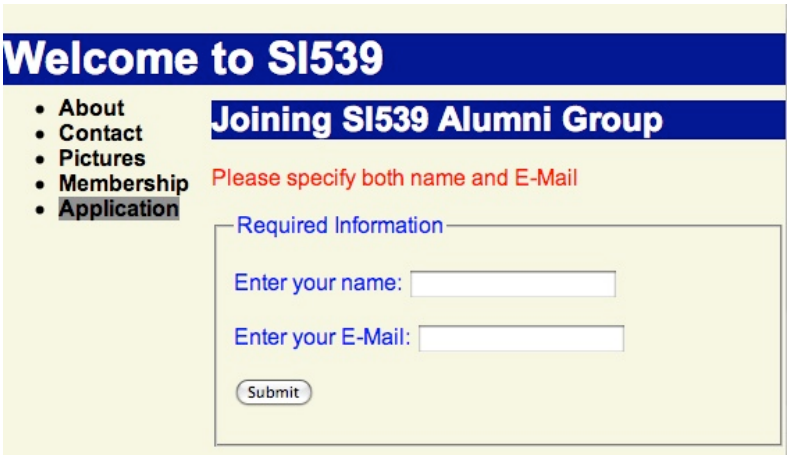
When a record is successfully deleted - you should see the flash message:




Your view page should look as follows:




Make sure to test that your join screen refuses to add a new person when the fields are left blank.




Make sure to look at your log and see that the helper is being called properly:

```
Processing OneController#join (for 127.0.0.1 at 2008-03-15 12:53:37) [GET]
  Session ID: b63cb8932b4b9d09bbd0d0ae1b1ad615
  Parameters: {"action"=>"join", "controller"=>"one"}
```

```
Rendering  within layouts/application
Rendering one/join
In do_select text=About action=index
urltext=/one
In do_select text=Contact action=contact
urltext=/one/contact
In do_select text=Pictures action=pictures
urltext=/one/pictures
In do_select text=Membership action=members
urltext=/one/members
In do_select text=Application action=join
We found the matching action!
urltext=/one/join
Rendered one/_nav (0.00221)
Completed in 0.00545 (183 reqs/sec) | Rendering: 0.00420 (77%) | 200 OK
[http://localhost/one/join]
```

You see that for every single page that includes navigation - the helper is called for each <li> entry and that one of the actions match.

**Hand In**

Hand in equivalent screen shots for you application for each of the screen shots shown above in this hand out.

In addition, hand in the following files:  one_controller.rb, index.rhtml, _nav.rhtml, one_helper.rb

If your program has a controller with a different name - just hand in the equivalent files.